## REMARKS/ARGUMENTS

Claims 1, 3-10, 12-19, and 21-27 are pending in the application. Claims 1, 10, and 19 have been amended. Reconsideration is respectfully requested. Applicants submit that the pending claims 1, 3-10, 12-19, and 21-27 are patentable over the art of record and allowance is respectfully requested of claims 1, 3-10, 12-19, and 21-27.

Applicants would like to thank Examiner Kiss for holding a telephone conference with their representative, Janaki K. Davda, on February 24, 2004. During the telephone conference, a minor proposed amendment adding "by" to claim 1 was discussed, and Examiner Kiss indicated that this minor amendment would be entered for purposes of appeal.

Additionally, in paragraph 3, the Specification is objected to due to the use of various trademarks. Applicants' have amended the Specification to overcome the objection.

In paragraph 5, claims 1-4, 6, 10-13, 15, 19-22, and 24 are rejected under 35 U.S.C. 102(b) as being anticipated by Ron Petrusha, "Inside the Windows 95 Registry" (hereinafter Petrusha).

Claims 1, 10, and 19 describe that a command is received from an application program for at least one variable maintained by the operating system. It is determined whether the at least one variable is in a data object. See, for example, Applicants' Specification at page 12, lines 26-27, and FIG. 3a, block 402. If the at least one variable is in the data object, the at least one variable is returned to the application program. See, for example, Applicants' Specification at page 12, lines 27-29, and FIG. 3a, blocks 404-406.

If the at least one variable is not in the data object, then the command from the application program is executed to retrieve and store the at least one variable in the data object. In particular, an operating system native command to use to retrieve the at least one variable is determined. For example, claims 5, 14, and 23 describe that the operating system native

command is selected from a set of native operating system commands for different types of operating systems, wherein the application program is capable of executing on each of the different types of operating systems. For example, see Applicants' Specification, page 13, lines 2-5 and FIG. 3a, blocks 408-418; and page 14, lines 12-28. Thus, for different operating systems, the same command from an application program will cause different operating system native commands to be executed to retrieve a variable. Also, Applicants' Specification describes that a cross-platform program is capable of accessing operating system information using one of many available native operating system commands. For example, see Applicant' Specification, page 14, lines 12-28.

The operating system native command is executed in response to the command from the application program to retrieve the at least one variable into a buffer. For example, see Applicants' Specification, page 13, lines 11-13. The retrieved at least one variable is stored from the buffer into the data object. For example, see Applicants' Specification at page 13, lines 14-17, which indicates that environment variables are generated as a data stream, and this output data stream is captured and read to obtain the content of the variables. Applicants' Specification at page 13, lines 27-28, indicates that the content of the variables is added to the data object. The command from the application program is executed to retrieve the at least one variable from the data object for return to the application program. For example, see Applicants' Specification, page 14, lines 7-11 and FIG. 3b, block 440.

The Office Action cites the Petrusha reference on page 35 pages 38-41 as teaching the subject matter of claims 1, 10, and 19. The cited portion of the Petrusha reference describes a registry, which is a database, and a Registry Editor ("RegEdit") that provides a user interface for browsing the registry. Also, at pages 61-68, the Petrusha reference describes that RegEdit depends on the registry functions within the Win32 API to gather information (page 61).

The Office Action indicates that the claimed elements of determination of an operating system native command to retrieve at least one variable, execution of the operating system native command in response to the command from an application program, and storing the retrieved at

least one variable from the buffer into a data object are disclosed by the registry editor interacting with the operating system's registry to retrieve data through the WIN32 Registry API. Applicants traverse. Unlike the technique described in the Petrusha reference, with Applicants' claimed invention, for different operating systems, *the same command from an application program will cause different operating system native commands to be executed to retrieve a variable.* Thus, with Applicants' claimed invention an operating system native command is determined for the command from the application program. On the other hand, with the Petrusha reference, the same registry function is used to retrieve a variable, and there is no need to determine an operating system native command. Thus, the Petrusha reference teaches away from the claimed subject matter.

The Office Action submits that the Registry Editor is an application program. Applicants traverse and submit that the Registry Editor is an operating system tool. For example, Applicants are submitting with this amendment, as Appendix A, a document printed from http://support.microsoft.com/default.aspx?scid=kb;en-gb;835823, retrieved on March 8, 2004, which indicates that "Windows comes with a tool called the Registry Editor for making changes" (see "How Can I Access the Registry to Make Changes to it?" on first page of document). Because the Windows 95 editor comes with the Registry Editor tool, there is no need in the Petrusha reference to determine an operating system native command. In particular, the registry functions within the Win32 APIs invoked by the Registry Editor are executed without further determining an operating system native command.

Moreover, in the claimed invention, the command from the application program is executed to retrieve the variable from the data object for return to the application program. The Office Action indicates that executing the command from the application program to retrieve the at least one variable from the data object for return to the application program is anticipated with "displaying the tree structure." Applicants traverse. Displaying a tree structure does not anticipate executing the command from the application program to retrieve the at least one variable from the data object for return to the application program.

Also, the Petrusha reference describes that the RegEdit user interface includes a left-hand pane (a "key pane") that displays registry keys and subkeys (page 38). The key pane of the RegEdit user interface lets a user know which registry keys have unexpanded children and which have expanded children (page 64). Moreover, the Petrusha reference describes a ReqQueryInfoKey function that determines the length of a registry key's longest subkey name to prevent allocating too small a buffer for rettieving subkey names, and a RegEnumKeyEx function that retrieves the name of the first subkey belonging to each top-level key retrieving values (page 65). Although the Petrusha references describes a buffer and data in the buffer being returned to the RedEdit program, the Petrusha reference does not describe storing data from the buffer into a data object.

The Office Action indicates that the Petrusha reference's description that if a selected key's subkey information has not yet been gathered by the program, the subkeys are retrieved, enumerated, and added as nodes in the TreeView control (page 68) as disclosing determining whether the requested variable is in the data object, wherein the command from the application program is executed to store at least one variable maintained by the operating system in the data object accessible to the application program. Claim 1 describes determining whether the at least one variable is in a data object, if the at least one variable is in a data object, returning the at least one variable to the application program, and, if the at least one variable is not in the data object, executing the command from the application program to store at least one variable maintained by the operating system in the data object accessible to the application program, wherein the application program is executing on the operating system. The Petrusha reference does not describe the claimed data object, into which data from a buffer is later stored. Furthermore, the description of Petrusha's reference that if a selected key's subkey information has not yet been gathered by the program, the subkeys are retrieved, enumerated, and added as nodes in the TreeView control does not anticipate if the at least one variable is in a data object, returning the at least one variable to the application program, and, if the at least one variable is not in the data object, executing the command from the application program to store at least one variable

maintained by the operating system in the data object accessible to the application program, wherein the application program is executing on the operating system.

Therefore, claims 1, 10, and 19 are not anticipated by the Petrusha reference. Dependent claims 3-4, 6, 11-13, 15, 20-22, and 24 incorporate the language of independent claims 1, 10, and 19, respectively, and add additional novel elements. Therefore, dependent claims 3-4, 6, 11-13, 15, 20-22, and 24 are not anticipated by the Petrusha reference.

In paragraph 12, the Office Action rejects claims 5, 9,4, 18, 23, and 27 under 35 U.S.C. 103(a) as being unpatentable over Petrusha as applied to claims 1, 8, 10, 17, 19, and 26.

The Office Action indicates that pages 183-206 discuss various platforms on pages 183-206, but the Petrusha reference in these pages describe one platform, a Windows95 platform, and different programs, Win16 and DOS programs that run under same Windows 95 platform and can access the registry (page 183). For example, DOS programs running under the Windows 95 platform access the registry with parameter validation, by pushing parameters on the stack, and moving function numbers into AX, and calling the VMM (page 191). Thus, the different programs are running on the same Windows 95 platform, so there is no need to select a native operating system command.

Claims 5, 14, and 23 describe determining a type of the operating system. The Office Action submits that the last paragraph of page 186 and code examples on page 187 and pages 521-629 teach determining a type of operating system with the expressed motivation of knowing which platforms (Windows 3.1, Windows 95, and Windows NT) a registry-enabled application is running on in order to allow for compensation for differences in the registry APIs and the registries themselves, and the Petrusha reference at these pages describes that since the registries themselves are so different in Windows 3.1, Windows 95, and Windows NT, it is important to know which platform the registry enabled application is running on. Applicants traverse. Although a type of operating system is determined, there is no teaching or suggestion that an operating system native command is selected in response to a command received from an

application program. Instead, because each operating system includes a Registry Editor tool, there is no need to determine a type of operating system to execute a Registry Editor function.

Moreover, with reference to claims 5, 14, and 23 the Office Action indicates that the Petrusha patent fails to expressly disclose selecting the operating system native command from a set of native operating system commands for different types of operating systems, wherein the application program is capable of executing on each of the different types of operating systems, but that one of ordinary skill in the art would recognize that because of the amount of backward compatibility built into various Microsoft® Windows® platforms it has been well known to have applications capable of running on multiple platforms and that it would have been obvious to include the selection of operating system commands to access environment variables (see page 10 of Office Action). Applicants traverse. Because the Registry Editor is a tool that is part of an operating system, each different operating system would include its own version of the Registry Editor. Therefore, there would be no need to determine the operating system native command from a set of native operating system commands for different types of operating systems, wherein the application program is capable of executing on each of the different types of operating systems.

As to claims 9, 18, and 27, the Office Action takes Official Notice that it has been known to employ line wrapping text files when a line exceeds a predetermined length.

Claims 5, 9, 14, 18, 23, and 27 incorporate the language of independent claims 1, 10, and 19, respectively, and add additional novel elements. Therefore, dependent claims 5, 9, 14, 18, 23, and 27 are not taught or suggested by the Petrusha reference.

In paragraph 6, the Examiner submits that Applicant must cite portions of the applied references that teach any express reasoning against or undesirable results of employing Applicants' claimed features to argue that a reference teaches against. Applicants traverse. "A reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or *would be led in a*

*direction divergent from the path that was taken by the applicant* . . . [or] if it suggests that the line of development flowing from the reference's disclosure is unlikely to be productive of the result sought by the applicant." In re Gurley, 27 F.3d 551, 553, 31 USPQ2d 1130, 1131 (Fed Cir. 1994) [emphasis added]. Each of the previously cited references taught a technique which provided an alternate path to obtaining data than Applicants' technique. Thus, one reading the previously cited references would have been lead in a direction divergent from the path that was taken by the application.

Conclusion

For all the above reasons, Applicant submits that the pending claims 1, 3-10, 12-19, and 21-27 are patentable over the art of record. Applicants have not added any claims. Nonetheless, should any additional fees be required, please charge Deposit Account No. 09-0447.

The attorney of record invites the Examiner to contact her at (310) 553-7973 if the Examiner believes such contact would advance the prosecution of the case.

Dated: April 15, 2004

By: Janaki K. Davda

Janaki K. Davda
Registration No. 40,684

Please direct all correspondences to:

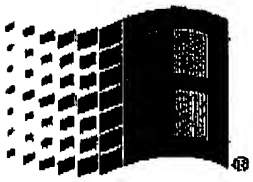
David Victor
Konrad Raynes & Victor, LLP
315 South Beverly Drive, Ste. 210
Beverly Hills, CA 90212
Tel: 310-553-7977
Fax: 310-556-7984

Appendix A

Microsoft Knowledge Base Article - 835823

# Microsoft Knowledge Base Article - 835823

## Registry problems





The Registry lies at the heart of Windows, controlling how your OS interacts with your hardware and applications. So what do you do when it starts to play up?

### Getting Started

The Windows Registry is a database of settings for all the hardware, software and different user preferences on your PC. Whenever you add hardware, uninstall a program or change the resolution of your display, it's the Registry that holds the data about this change.

It works away in the background and is not designed to be accessed or edited by the casual Windows user. This is because making an incorrect entry in the Registry can cause major problems, including your PC no longer starting. For this reason you shouldn't start tweak Registry settings unless you're confident about the changes you're making and have a backup plan in place. That said, there are times when being able to access and edit the Registry can be incredibly handy, healing a sick PC in minutes. We're going to show you how to access and back up your Registry, plus solve common problems with a simple setting change.

### How can I access the Registry to make changes to it?

The Registry data is contained within a handful of files, depending on the Operating System you use. For example, in Windows 98 there are two hidden files called User.dat and System.dat in your Windows directory. However, you can't open and edit these files directly and Windows comes with a tool called the Registry Editor for making changes. You can access this tool in all versions of Windows by pressing Start, Run, typing regedit and pressing OK.

### How do I go about backing up the Registry?

Understanding whether you have a good backup and how to restore it if things go wrong is one of the most important questions you should address before attempting to make any changes. The process varies according to your version of Windows.

For Windows 98 press Start, Run, type scanregw and click OK. When you receive a prompt to back up the Registry click Yes and then press OK when you're informed that the process is complete.

### My Windows 98 Registry is working OK, but the associated files have grown massively and there are loads of redundant entries in there.

The longer you run a PC without a clean reinstall, the more your Registry files grow. This is just a fact of Windows and over time it can cause your PC to slowdown. While Windows is good at adding and tracking changes in your Registry, tasks such as uninstalling software do not always cause the Registry entries to be removed. There's a utility for Windows 98 called RegClean that can help you get rid of all these erroneous entries and speed up your PC in the process. You can download it from http://download.com.com/3000-2094-881470.html.

### How do I create new Keys, String values, or edit value data in the Registry Editor tool?

Just browse to the key you're interested in. If you want to create a new subkey for this key, or a new value within this key, then right-click it and select New. You can then choose to create a new key or new value type as required. To edit value data, browse to the value and simply double-click it. If the data can be edited then a box will appear that you can simply type the new data into.

---

### Sometimes when I click on my mouse, it acts as if I have clicked it twice, rather than once.

There is a setting in the Registry that you can switch on to detect accidental double mouse clicks. Browse to HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced, create a new String value called UseDoubleClickTimer and set the value data to 1.

---

### My Windows text is difficult to read, with jagged edges on the letters. I have checked that the fonts are installed correctly. What can I do?

A Registry setting can smooth the fonts displayed on your PC by using anti-aliasing. It's set on by default, so it may be that this has value has been modified, or deleted. Browse to HKEY_CURRENT_USER\Control Panel\Desktop and see if the String FontSmoothing is present. If it's there set the value to 2. If it isn't, recreate it.

---

### I love the thumbnail view in Explorer, but is there a way that I can make the pictures easier to see?

Browse to HKEY_CURRENT_ USER\Software\Microsoft\Windows\CurrentVersion\Explorer. Create or modify two DWORDs called ThumbnailSize and ThumbnailQuality. For ThumbnailSize set the value in pixels, with the default being 96. For ThumbnailQuality set the value as a number which represents the percentage quality between 50 and 100.

---

### I have programs still listed in the Add/Remove programs dialog even though they have already been uninstalled. How do I get rid of them?

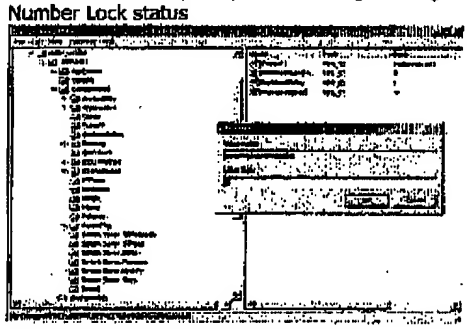
The Registry can help you get rid of these redundant entries. Browse to the following key: HKEY_ LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall.

You will find a collection of subkeys beneath this key, each one representing an entry in the Add/Remove programs dialog. Click any subkey to see the program that it is associated with, which can be viewed in the DisplayName value. Delete the subkey to remove the entry.

---

### A simple Registry tweak

To understand the principles of altering the Registry, follow this simple tweak for changing the Number Lock status



Press Start/Run and type regedit to launch the Registry Editor. Use the Explorer-style browse tree in the left-hand pane to find HKEY_USERS\.DEFAULT\Control Panel\Keyboard.

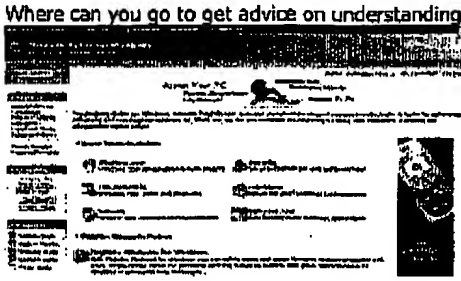
Look in the right-hand pane for the value InitialKeyboardIndicators and double-click it. In the Edit String dialog box change the Value data to either 0 for Off or 2 for On. Press OK and close

the Registry Editor.

The change made will apply to all users, but if you want to make the change just for yourself and not for other users on the PC then you can do this by making the same changes to the following key: HKEY_CURRENT_USER\Control Panel\Keyboard.

**REGISTRY HELP**

Where can you go to get advice on understanding and modifying the Registry?



If you find yourself in need of help and we're not at hand, there's one online source that stands head and shoulders above the rest.

Formerly known as Regedit.com, the Windows Registry Guide can be found at www.regedit.com. You'll find detailed tutorials explaining what the Registry is and how it works. There are links to a host of utilities that can help you make the most of your Registry and keep it fine tuned. Most important of all, you'll find just about every tweak you'll ever need, divided into helpful sections. These range from fixing problems with hardware and software, through to tips and tricks that will optimise your system.

The Registry is one of the most powerful components of your Operating System and the Windows Registry Guide can help you unlock its full potential. You can even access the contents of the guide when you're not online by downloading and installing a small application from www.winguides.com/guides.php?guide=registry



**The information in this article applies to:**

- the operating system: Microsoft Windows 98
- the operating system: Microsoft Windows 98 Second Edition
- Microsoft Windows 98 Second Edition

**Last Reviewed:** 28/01/2004 (1.0)